ENKBG·1.0 VAX 11/730 APT/RD LOOPBACK TST

EP-ENKBG-DL-1.0 FICHE 1 OF 1

JUL 1982
COPYRIGHT© 1982
MADE IN USA

d i g i t a l

IDENTIFICATION

Product code: ENKBG VERSION 1.00

Product name: VAX-11/730 APT/RD USART MODEM LOOPBACK TEST

Product date: 30-APR-1982

Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1982 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

DEC	DECsystem-10	DECSYSTEM-20
DECUS	MASSBUS	PDP
UNIBUS	VAX	VMS

digital

Table of Contents

1.0 ABSTRACT 3

2.0 HARDWARE REQUIREMENTS 3

3.0 SOFTWARE REQUIREMENTS 3

4.0 PREREQUISITES 3

5.0 OPERATING INSTRUCTIONS 4
5.1 Options 4
5.2 Event Flags 4
5.3 APT Setup 4

6.0 PROGRAM FUNCTIONAL DESCRIPTION 4
6.1 Program Overview 4
6.2 Program Size 4
6.3 Program Run Times 5
6.4 Run-time Dynamics 5
6.5 Fault Detection 5
6.6 Performance During Hardware Failures 5
6.7 Program Applications 6
6.8 Test Descriptions 6

7.0 MAINTENANCE HISTORY 6

1.0 ABSTRACT

This program is an 8085 based micro-diagnostic designed to test the APT/RD usart hardware on the WCS (Writable Control Store) module of the VAX-11/730 processor.

2.0 HARDWARE REQUIREMENTS

This program will run with a minimum hardware configuration of the WCS and CPU modules present. It also requires the use of either a modem in loopback mode or a turnaround connector on the APT/RD Usart.

Other hardware modules and options may be installed without affecting the diagnostic's performance.

3.0 SOFTWARE REQUIREMENTS

This diagnostic must be run under the VAX-11/730 micro monitor (ENKAA) in a standalone environment. The micro-diagnostic is written into the top 3K of 8085 RAM. The micro-diagnostic monitor also resides in 8085 ram, where console software is normally present.

4.0 PREREQUISITES

This module should be run after running modules ENKBA thru ENKBD. It thereby assumes that the Usart internal loopback selftests have successfully completed.

This program also assumes that there is some sort of wraparound capability enabled on the APT-RD USART. This could either mean a modem set in wraparound mode, or a turnaround connector attached to the port cable.

5.0 OPERATING INSTRUCTIONS

This program may be executed by typing DI SE ENKBG after the micro-monitor has been loaded and the MI> prompt has been printed at the terminal. Both the micro-monitor and this program are loaded from a TU58 cassette or downline loaded through the APT-RD port. See the OVERVIEW MANUAL for more information.

5.1 Options

Not applicable

5.2 Event Flags

Not applicable

5.3 APT Setup

Unlike other VAX-11/730 micro-diagnostics, ENKBG can not be run under APT. Though the program may be downline loaded through the APT-RD port like other micro-diagnostics, because it requires some kind of turnaround on the port, it can never be run successfully under APT.

6.0 PROGRAM FUNCTIONAL DESCRIPTION

6.1 Program Overview

This program is designed to detect stuck-at faults and provide a repair tool which helps isolate the failing component. Though the APT-RD USART drivers and receivers are the most likely sources of error, this program will also detect faults in the cable, connectors, and the inversion logic that is between the APT-RD USART and the connector.

6.2 Program Size

This program runs in 8085 RAM memory and is approximately 512 bytes long.

6.3 Program Run Times

This program will take approximately 10 seconds to run.

6.4 Run-time Dynamics

Not applicable

6.5 Fault Detection

The diagnostic will report errors with the following header:

```
SECT  TST ERR   EXP   REC  OTHER  MSK  MODULE
```

1. SECT is the program name (ENKBG)
2. TST is the test number that failed.(Test 2F is the only test.)
3. ERR is the error number that failed.
4. EXP is the expected (correct) data.
5. REC is the received (actual) data.
6. OTHER is any other pertinent data (such as an address). This field is not always used. N/A will be printed under OTHER when not in use. The ERRORS section in the listing will describe the contents of this field when it is used.
7. MASK is the error mask used to check the result. Bits set to a 1 in this mask correspond to bits in the result that are not checked.
8. MODULE is the suspected module that caused the failure.

6.6 Performance During Hardware Failures

A power fail will cause a rebooting of the system (if a battery backup is not present). Other hardware failures will cause normal error printouts.

6.7 Program Applications

This program can be used by field service to determine which module should be replaced and to verify that the replacement eliminated the failure. It can also be used as a repair tool to help isolate a failing component by manufacturing, field service or engineering. The program is NOT APT compatible.

Customers may use the program to verify the basic integrity of the APT-RD port on the WCS module in the system.

6.8 Test Descriptions

See the actual test in the listing for detailed descriptions and error analyses. A brief description of the logic being tested by each test can be found in the table of contents of the listing.

7.0 MAINTENANCE HISTORY

DATE	VERSION	DESCRIPTION
30-APR-82	1.0	Initial release.

```
ZZ-ENKBG-1.0    7000    4
7000     3 .TITLE  'ENKBG  VAX-11/730 APT/RD USART MODEM LOOPBACK TEST Rev 01.00'
7000     4 ;
7000     5 ;                                  COPYRIGHT (c) 1982 BY
7000     6 ;                          DIGITAL EQUIPMENT CORPORATION, MAYNARD,
7000     7 ;                           MASSACHUSETTS.  ALL RIGHTS RESERVED.
7000     8 ;
7000     9 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY  BE  USED  AND  COPIED
7000    10 ; ONLY  IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION
7000    11 ; OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY OTHER COPIES THEREOF
7000    12 ; MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER  PERSON.  NO
7000    13 ; TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.
7000    14 ;
7000    15 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE, AND
7000    16 ; SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
7000    17 ;
7000    18 ; DIGITAL  ASSUMES  NO  RESPONSIBILITY  FOR  THE  USE OR RELIABILITY OF ITS
7000    19 ; SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.
7000    20 ;
7000    21 ;
7000    22 ;++
7000    23 ; FACILITY:     VAX-11/730 MICRO-DIAGNOSTIC.
7000    24 ;
7000    25 ; ABSTRACT:     This 8085 based diagnostic tests the hardcore logic of the
7000    26 ;                APT/RD USART of the WCS module.
7000    27 ;
7000    28 ; ENVIRONMENT:  ENKAA  Micro-diagnostic Monitor
7000    29 ;
7000    30 ; AUTHOR:       David Mayo      27-APR-1982     VERSION 01.00
7000    31 ;
7000    32 ; MODIFIED BY:
7000    33 ;--
7000    39
7000    40
7000    41
7000    42
7000    43
7000    44
7000    45
7000    46
7000    47
7000    48
7000    49
7000    50
7000    51
7000    52
7000    53
7000    54
7000    55
7000    56
7000    57
7000    58
7000    59
7000    60
7000    61
7000    62
7000    63
7000    64
7000    65
7000    66
7000    67
```

```
                                .PSECT  CPU6,BYTE
                                .ALIGN  BYTE

                        ;*******************************************************************************
                        ;
                        ; DATA AREA
                        ;
                        ;*******************************************************************************

7000    78 00000000     HEAD            =0
7000    79
7000    80 00000000                     EQUATE                  ;REGISTER EQUATE MACRO
           00000002
           00000004
           00000006
           00000006
           00000001
           00000003
           00000005
           00000007 00000006
7000    81
7000    82 00000000     UPC14A          =<^X00>                 ;(RO) UPC BIT 14 ASSERTED HIGH <MSB>
7000    83
7000    84
7000    85 00000001     MISC2           =<^X01>
7000    86 00000001     BOOTF           =<^X01>                 ;(RO) BOOT FLAG. ASSERTED LOW. <LSB>
7000    87
7000    88
7000    89 00000002     DCLO            =<^X02>                 ;(RO) DC LOW FLAG ASSERTED HIGH<MSB>
7000    90
7000    91 00000002     HLTFLG          =<^X02>                 ;(RO) HALT FLAG ASSERTED LOW   <LSB>
7000    92
7000    93
7000    94 00000003     READJ1          =<^X03>                 ;(RO) JUMPER J1 FLAG ASS. HIGH <MSB>
7000    95
7000    96 00000003     ALLOWP          =<^X03>                 ;(RO) ENABLE CONTROL P ASS LOW <LSB>
7000    97
7000    98
7000    99 00000004     READJ2          =<^X04>                 ;(RO) BIT 07 EQUALS J2 <MSB>
7000   100 00000004     APTFLG          =<^X04>                 ;(RO) APT PRESENT FLAG <LSB>
7000   101
7000   102 00000005     AUTO            =<^X05>                 ;(RO) AUTO TEST FLAG ASS LOW <LSB>
7000   103
7000   104 00000006     REMDIS          =<^X06>                 ;(RO) REMOTE DISABLE FLAG ASS LOW <LSB>
7000   105
7000   106 00000007     OKV15           =<^X07>                 ;(RO) 15 VOLTS OK FLAG ASS LOW <MSB>
7000   107
7000   108 00000007     BOOTEN          =<^X07>                 ;(RO) BOOT ENABLE FLAG ASS LOW <LSB>
7000   109
7000   110
7000   111 00000008     WCSB0           =<^X08>                 ;(WO) WCS WRITE DATA REGISTER
7000   112 00000009     WCSB1           =<^X09>
7000   113 0000000A     WCSB2           =<^X0A>
7000   114
7000   115 0000001C     ITDB            =<^X1C>                 ;(R/W) TIMER DATA REGISTER
7000   116 0000001D     ITCSR           =<^X1D>                 ;(R/W) TIMER STATUS REGISTER
7000   117
7000   118 00000020     CLRCLK          =<^X20>                 ;(WO) CLEAR CPU CLOCK RUN
7000   119 00000021     SETCLK          =<^X21>                 ;(WO) SET CPU CLOCK RUN
```

```
7000  120
7000  121 00000022                    CLRSS       =<^X22>     ;(WO) CLEAR CLOCK SINGLE STEP
7000  122 00000023                    SETSS       =<^X23>     ;(WO) SET CLOCK SINGLE STEP
7000  123
7000  124 00000024                    CLRCSK      =<^X24>     ;(WO) CLEAR CSR CLOCK
7000  125 00000025                    SETCSK      =<^X25>     ;(WO) SET CSR CLOCK
7000  126
7000  127 00000026                    CLRUPK      =<^X26>     ;(WO) CLEAR THE UPC CLOCK
7000  128 00C00027                    SETUPK      =<^X27>     ;(WO) SET THE UPC CLOCK
7000  129
7000  130 00000028                    SETMCI      =<^X28>     ;(WO) SET MEMORY CONTROL INIT
7000  131 00000029                    CLRMCI      =<^X29>     ;(WO) CLEAR MEMORY CONTROL INIT
7000  132
7000  133 0000002A                    SETMCK      =<^X2A>     ;(WO) SET MEMORY CONTROL CLOCK
7000  134 0000002B                    CLRMCK      =<^X2B>     ;(WO) CLEAR MEMORY CONTROL CLOCK
7000  135
7000  136 0000002C                    CLRTMR      =<^X2C>     ;(WO) STOP TIMER
7000  137 0000002D                    SETTMR      =<^X2D>     ;(WO) START TIMER COUNTING
7000  138
7000  139 0000002E                    CLRBSY      =<^X2E>     ;(WO) CLEAR UNIBUS BUS BUSY
7000  140 0000002F                    SETBSY      =<^X2F>     ;(WO) SET UNIBUS BUS BUSY
7000  141
7000  142 00000030                    SETDCL      =<^X30>     ;(WO) SET DC LOW
7000  143 00000031                    CLRDCL      =<^X31>     ;(WO) CLEAR DC LOW
7000  144
7000  145 00000032                    SETACL      =<^X32>     ;(WO) SET AC LOW
7000  146 00000033                    CLRACL      =<^X33>     ;(WO) CLEAR AC LOW
7000  147
7000  148 00000034                    INITL       =<^X34>     ;(WO) CLEAR UNIBUS INIT
7000  149 00000035                    INITH       =<^X35>     ;(WO) SET UNIBUS INIT
7000  150
7000  151 00000036                    CLRWCK      =<^X36>     ;(WO) CLEAR WCS BIT 00
7000  152 00000037                    SETWCK      =<^X37>     ;(WO) SET WCS BIT 00
7000  153
7000  154 00000038                    CLRCSR      =<^X38>     ;(WO) CLEAR WCS BIT 01
7000  155 00000039                    SETCSR      =<^X39>     ;(WO) SET WCS BIT 01
7000  156
7000  157 0000003C                    CLRLIT      =<^X3C>     ;(WO) CLEAR THE RUN LIGHT
7000  158 0000003D                    SETLIT      =<^X3D>     ;(WO) SET THE RUN LIGHT
7000  159
7000  160 00000040                    TRDB        =<^X40>     ;(R/W) REMOTE USART DATA BUFFER
7000  161 00000041                    TRSR        =<^X41>     ;(RO)  REMOTE USART STATUS REGISTER
7000  162 00000042                    TRMODE      =<^X42>     ;(R/W) REMOTE MODE REGISTER 1 + 2
7000  163                                                      ;FIRST ACCESS GET MR1, SECOND GETS MR2
7000  164                                                      ;READING THE CR FORCES THE INTERNAL
7000  165                                                      ;POINTER BACK TO MR1
7000  166 00000043                    TRCR        =<^X43>     ;(R/W) REMOTE USART COMMAND REGISTER
7000  167
7000  168 00000044                    TUDB        =<^X44>     ;(R/W) TU-58 DATA BUFFER
7000  169 00000045                    TUSR        =<^X45>     ;(RO)  TU-58 STATUS REGISTER
7000  170 00000046                    TUMODE      =<^X46>     ;(R/W) TU-58 MODE REGISTER 1 + 2
7000  171                                                      ;FIRST ACCESS GET MR1,SECOND GETS MR2
7000  172                                                      ;READING THE CR FORCES THE INTERNAL
7000  173                                                      ;POINTER BACK TO MR1
7000  174 00000047                    TUCR        =<^X47>     ;(R/W) TU-58 COMMAND REGISTER
7000  175
7000  176 00000048                    TTDB        =<^X48>     ;(R/W) TERMINAL DATA BUFFER
7000  177 00000049                    TTSR        =<^X49>     ;(RO)  TERMINAL STATUS REGISTER
7000  178 0000004A                    TTMODE      =<^X4A>     ;(R/W) TERMINAL MODE REGISTER 1 + 2
7000  179                                                      ;FIRST ACCESS GET MR1,SECOND GETS MR2
```

```
7000  180
7000  181
7000  182 0000004B                    TTCR            =<^X4B>         ;READING THE CR FORCES THE INTERNAL
                                                                      ;POINTER BACK TO MR1
                                                                      ;(R/W) TERMINAL COMMAND REGISTER
7000  183
7000  184 00000080                    READ            =<^X80>         ;(RO) CONSOLE READ REGISTER
7000  185
7000  186 00000082                    CPUACK          =<^X82>         ;(RO) ACKNOWLEDGE FROM CPU      <LSB>
7000  187
7000  188 00000083                    CPATTN          =<^X83>         ;(RO) ATTENTION FROM CPU        <LSB>
7000  189
7000  190 00000084                    UPC14           =<^X84>         ;(RO) UPC BIT 14                <LSB>
7000  191
7000  192 00000085                    CSR07           =<^X85>         ;(RO) CSR BIT 7                 <LSB>
7000  193
7000  194 00000086                    CSR15           =<^X86>         ;(RO) CSR BIT 15                <LSB>
7000  195
7000  196 00000087                    CSR23           =<^X87>         ;(RO) CSR BIT 23                <LSB>
7000  197
7000  198 000000A0                    ENBPAR          =<^XA0>         ;(WO) ENABLE STALL ON PARITY ERROR
7000  199 000000A1                    DISPAR          =<^XA1>         ;(WO) DISABLE STALL ON PARITY ERROR
7000  200
7000  201 000000A2                    VSHIFT          =<^XA2>         ;(WO) SET THE V-BUS TO SHIFT MODE
7000  202 000000A3                    VNORML          =<^XA3>         ;(WO) SET THE V-BUS TO NORMAL MODE
7000  203
7000  204 000000A4                    SETTIM          =<^XA4>         ;(WO) SET THE INTERVAL TIMER INTERRUPT
7000  205 000000A5                    CLRTIM          =<^XA5>         ;(WO) CLEAR THE INTERVAL TIMER INTRPT.
7000  206
7000  207 000000A6                    ENBMEM          =<^XA6>         ;(WO) ENABLE MEMORY REFERENCES
7000  208 000000A7                    DISMEM          =<^XA7>         ;(WO) DISABLE MEMORY REFERENCES
7000  209
7000  210 000000A8                    SETACK          =<^XA8>         ;(WO) SET CONSOLE ACKNOWLEDGE
7000  211 000000A9                    CLRACK          =<^XA9>         ;(WO) CLEAR CONSOLE ACKNOWLEDGE
7000  212
7000  213                             ;NOTE:  THE CONSOLE ACKNOWLEDGE SIGNAL IS ALSO USED AS THE UPC
7000  214                             ;SERIAL INPUT.  WHEN USED THIS WAY THE SENSE IS INVERTED.  TO SIMPLIFY
7000  215                             ;MATTERS THE NEXT TWO EQUATES ARE INCLUDED
7000  216
7000  217 000000A8                    CLRUPC          =<^XA8>         ;(WO) CLEAR THE V-BUS SERIAL INPUT
7000  218 000000A9                    SETUPC          =<^XA9>         ;(WO) SET THE V-BUS SERIAL INPUT
7000  219
7000  220 000000AA                    SETATN          =<^XAA>         ;(WO) SET CONSOLE ATTENTION
7000  221 000000AB                    CLRATN          =<^XAB>         ;(WO) CLEAR CONSOLE ATTENTION
7000  222
7000  223 000000AC                    SETPFI          =<^XAC>         ;(WO) SET POWER FAIL INTERRUPT
7000  224 000000AD                    CLRPFI          =<^XAD>         ;(WO) CLEAR POWER FAIL INTERRUPT
7000  225
7000  226 000000AE                    SETHLT          =<^XAE>         ;(WO) SET THE CONSOLE HALT BIT
7000  227 000000AF                    CLRHLT          =<^XAF>         ;(WO) CLEAR THE CONSOLE HALT BIT
7000  228
7000  229 000000CC                    WRITE           =<^XCC>         ;(WO) THE CONSOLE WRITE REGISTER
7000  230
7000  231 000000EC                    YBUSRD          =<^XEC>         ;(RO) Y-BUS READ
7000  232
7000  233
7000  234                             ;ADDITIONAL DATA
7000  235
7000  236 00004C22                    PARITY_VECTOR   = <^X4C22>      ;PARITY ERROR BRANCH VECTOR.
7000  237 00004C25                    POWER_VECTOR    = <^X4C25>      ;POWER FAIL BRANCH VECTOR.
7000  238
7000  239                             ;USART DATA
```

```
7000  240
7000  241 00000002                TRSR_MASK       = <^X02>            ;MASK TO READ TR STATUS REGISTER
7000  242 00000037                TRCR_MASK       = <^X37>            ;MASK TO SET UP TR COMMAND REGISTER
7000  243
7000  244                         ;*****************************************************************************
7000  245                         ;
7000  246                         ;  PROGRAM ENTRY POINT.... MUST BE THE FIRST INSTUCTION IN THE PROGRAM
7000  247                         ;
7000  248                         ;*****************************************************************************
7000  249
7000  250
7000  251 00CC'C3                 ENTRY:          JMP     TEST2F
7003  252
7003  253                         ;*****************************************************************************
7003  254                         ;
7003  255                         ;  VERSION NUMBER AND LAST TWO LETTERS OF FILE NAME.  THESE 2 WORDS CAN NOT
7003  256                         ;  BE MOVED TO ANY OTHER LOCATION WITHOUT A MICMON CHANGE.
7003  257                         ;
7003  258                         ;*****************************************************************************
7003  259
7003  260 0100                    VERSION:        .WORD           <^X0100>        ;VERSION NUMBER (REV 01.00)
7005  261 47 42                   FILE_NAME:      .ASCII          'BG'            ;LAST 2 CHARS OF FILE NAME
7007  262                         ;*****************************************************************************
7007  263                         ;
7007  264                         ; VARIABLES AREA
7007  265                         ;
7007  266                         ;*****************************************************************************
7007  267
7007  268
7007  269 00                      BACKGRND:       .BYTE           0               ;BACKGROUND OF 1'S OR 0'S
7008  270 00                      DATA_PAT:       .BYTE           0               ;SHIFTING DATA PATTERN
7009  271 00                      EXP_DAT:        .BYTE           0               ;EXPECTED DATA AREA
700A  272 FF                      ONE_DAT:        .BYTE           <^XFF>          ;DATA PATTERN OF ALL ONES
700B  273 00                      REC_DAT:        .BYTE           0               ;RECEIVED DATA AREA
700C  274 00                      ZERO_DAT:       .BYTE           0               ;DATA PATTERN OF ALL ZEROS
700D  275
700D  276
700D  277
700D  278                         ;*****************************************************************************
700D  279                         ;
700D  280                         ;  TEST2F_SUB   THIS ROUTINE IS USED BY TEST 2F TO WRITE AN 8 BIT TEST PATTERN
700D  281                         ;               TO THE APT/RD UART AND THEN READ IT BACK
700D  282                         ;
700D  283                         ;
700D  284                         ;*****************************************************************************
700D  285 0009'32                 TEST2F_SUB:     STA     EXP_DAT         ;LOAD THE DATA PATTERN INTO EXP_DAT
7010  286
7010  287 A8 D3                   1$:             OUT     SETACK          ;SET THE SCOPE SYNC
7012  288 A9 D3                                   OUT     CLRACK          ;CLEAR IT
7014  289
7014  290 41 DB                                   IN      TRSR            ;LOOK FOR XMIT RDY
7016  291 1F                                      RAR
7017  292 0038'DA                                 JC      3$              ;CONTINUE IF XMIT RDY SET
701A  293
701A  294 00 FF 01                                LXI     B,<^X00FF>      ;WAIT HERE
701D  295 0B                    2$:             DCX     B
701E  296 78                                    MOV     A,B
701F  297 B1                                    ORA     C
7020  298 001D'C2                               JNZ     2$
7023  299
```

```
7023  300 41 DB                                    IN      TRSR            ;THEN LOOK FOR XMIT RDY AGAIN
7025  301 1F                                       RAR
7026  302 0038'DA                                  JC      3$              ;CONTINUE IF XMIT RDY SET
7029  303
7029  304 01 3E                                    MVI     A,1             ;ELSE REPORT ERROR 1
702B  305 0000'32                                  STA     CON_ERR_NUM
702E  306 0083'CD                                  CALL    TEST2F_ERR1_2   ;PRINT AN ERROR MESSAGE
7031  307
7031  308 0000'3A                                  LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7034  309 0F                                       RRC
7035  310 0010'DA                                  JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP
7038  311
7038  312 0009'3A                          3$:     LDA     EXP_DAT         ;GET THE PATTERN
703B  313
703B  314 40 D3                                    OUT     TRDB            ;SEND IT OUT
703D  315
703D  316 02 06                                    MVI     B,TRSR_MASK
703F  317
703F  318 41 DB                                    IN      TRSR            ;GET THE SR
7041  319 A0                                       ANA     B               ;MASK OUT ALL BITS EXCEPT REVR DONE
7042  320 0065'C2                                  JNZ     5$              ;CONTINUE IF REVR DONE SET
7045  321
7045  322 3F FF 01                                 LXI     B,<^X3FFF>      ;WAIT HERE
7048  323 0B                               4$:     DCX     B
7049  324 78                                       MOV     A,B
704A  325 B1                                       ORA     C
704B  326 0048'C2                                  JNZ     4$
704E  327
704E  328 02 06                                    MVI     B,TRSR_MASK
7050  329 41 DB                                    IN      TRSR            ;THEN LOOK FOR REVR DONE AGAIN
7052  330 A0                                       ANA     B
7053  331 0065'C2                                  JNZ     5$              ;CONTINUE IF REVR DONE SET
7056  332
7056  333 02 3E                                    MVI     A,2             ;ELSE REPORT ERROR 2
7058  334 0000'32                                  STA     CON_ERR_NUM
705B  335 0083'CD                                  CALL    TEST2F_ERR1_2   ;PRINT AN ERROR MESSAGE
705E  336
705E  337 0000'3A                                  LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7061  338 0F                                       RRC
7062  339 0010'DA                                  JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP
7065  340
7065  341 40 DB                            5$:     IN      TRDB            ;GET THE DATA
7067  342 000B'32                                  STA     REC_DAT         ;STORE DATA IN REC_DAT
706A  343
706A  344 00'09'21                         ******************              IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE EXPECTED AND RECEIVED
      345       01 0E 00'0B'11           *                                                                 ; DATA
      346       007B'CA 0000'CD          *
7078  347                                  *
7078  348 009B'CD                         *                                CALL    TEST2F_ERROR    ;PRINT AN ERROR MESSAGE
7078  349                                  *
707B  350                                  ******************              ENDIF
707B  351
707B  352 0000'3A                                  LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
707E  353 0F                                       RRC                     ;
707F  354 0010'DA                                  JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP
7082  355
7082  356 C9                                       RET
7083  357
```

```
7083  358
7083  359
7083  360
7083  361
7083  362
7083  363
7083  364
7083  365 0000'32 3C AF
7088  366 0009'3A
708B  367 0003'32
708E  368 0000'32 AF
7092  369 00 00 21
7095  370 0000'22
7098  371 00BD'C3
709B  372
709B  373 03 3E
709D  374 0000'32
70A0  375
70A0  376 0009'3A
70A3  377 0003'32
70A6  378
70A6  379 000B'3A
70A9  380 0003'32
70AC  381
70AC  382 0000'32 3C AF
70B1  383
70B1  384 FF FF 21
70B4  385 0000'22
70B7  386 00 FF 21
70BA  387 0002'22
70BD  388
70BD  389 00'00'21
70C0  390
70C0  391 00'00'11
70C3  392 05 0E
70C5  393 0000'CD
70C8  394
70C8  395 0000'CD
70CB  396
70CB  397 C9
70CC  398
70CC  399
70CC  400
70CC  401
70CC  402
70CC  403
70CC  404
70CC  405
70CC  406
70CC  407
70CC  408
70CC  409
70CC  410
70CC  411
70CC  412
70CC  413
70CC  414
70CC  415
70CC  416
70CC  417
```

```
;*****************************************************************************
;
;  TEST2F_ERROR  THIS ROUTINE IS USED BY TEST 2F TO DEFINE THE ERROR VARIABLES
;                IN PREPARATION FOR USING THE CON_ERROR ROUTINE
;
;*****************************************************************************

TEST2F_ERR1_2:  SET     NA_EXP_REC      ;EXPECTED AND RECEIVED DATA IS N/A
                LDA     EXP_DAT         ;MOVE EXPECTED DATA TO CON_ADD_DAT
                STA     CON_ADD_DAT+3
                CLEAR   NA_OTHER        ;OTHER DATA CONTAINS CURRENT PATTERN
                LXI     H,<^X0000>      ;ERROR MASK IS ZERO
                SHLD    CON_MSK_DAT
                JMP     REPORT_ERR

TEST2F_ERROR:   MVI     A,3             ;ERROR NUMBER 3
                STA     CON_ERR_NUM     ;

                LDA     EXP_DAT         ;MOVE EXPECTED DATA TO CON_EXP_DAT
                STA     CON_EXP_DAT+3

                LDA     REC_DAT         ;MOVE RECEIVED DATA TO CON_REC_DAT
                STA     CON_REC_DAT+3

                SET     NA_OTHER        ;OTHER DATA NOT APPLICABLE

                LXI     H,<^XFFFF>      ;ERROR MASK INCLUDES ONLY THE LEAST
                SHLD    CON_MSK_DAT     ; SIGNIFICANT BYTE
                LXI     H,<^X00FF>
                SHLD    CON_MSK_DAT+2

REPORT_ERR:     LXI     H,MWCS_A        ;WCS MODULE BEING TESTED

                LXI     D,CON_MOD_DAT   ;LOAD THE MODULE DATA POINT TO BY
                MVI     C,5             ; THE H,L PAIR INTO CON_MOD_DAT.
                CALL    MOVER

                CALL    CON_ERROR       ;CALL ERROR ROUTINE

                RET


;*****************************************************************************
;
;
;       TEST 2F         APT/RD USART MODEM LOOPBACK TEST
;       -------         --------------------------------
;
;               THE APT/RD UART MODEM LOOPBACK TEST WILL WRITE ALL 0'S,
;       ALL 1'S, 0 RIPPLED THROUGH A FIELD OF 1'S, AND 1 RIPPLED THROUGH
;       A FIELD OF 0'S TO THE APT/RD UART.  IT IS AN EXTENSION OF THE
;       UART INTERNAL LOOPBACK SELF TEST TO BE USED FOR APT AND FIELD SERVICE
;       PURPOSES.  THIS IS AN OPTIONAL TEST, ONLY RUN WHEN EXPLICITLY
;       SPECIFIED.  FOR THIS TEST TO RUN PROPERLY, EITHER A MODEM OR A
;       TURNAROUND CONNECTOR MUST BE CONNECTED TO THE APT/RD USART CABLE.
;
;       TEST STEPS:
;               1) SET UP APT/RD USART FOR TEST.
;               2) WAIT FOR XMIT RDY FROM USART.
;               3) ISSUE ERROR 1 OF XMIT RDY NOT SET.
```

```
ZZ-ENKBG-1.0    7000    4
```

```
                        4) WRITE ALL 0'S OUT TO THE USART.
                        5) WAIT FOR REVR DONE FROM USART.
                        6) ISSUE ERROR 2 IF REVR DONE NOT SET.
                        7) READ DATA FROM USART DATA BUFFER.
                        8) ISSUE ERROR 3 IF RECEIVED DATA IS NOT EQUAL TO
                           TRANSMITTED DATA.
                        9) REPEAT STEPS 2-8 FOR DATA OF ALL 1'S.
                       10) REPEAT STEPS 2-8 FOR 0 SHIFTED IN FIELD OF 1'S.
                       11) REPEAT STEPS 2-8 FOR 1 SHIFTED IN FIELD OF 0'S.

;ASSUMPTIONS:  IT IS ASSUMED THAT EITHER A MODEM WITH WRAPAROUND
;              CAPABILITIES ENABLED OR A TURNAROUND CONNECTOR IS
;              CONNECTED TO THE APT/RD UART.

;ERROR 1: XMIT RDY NEVER RECEIVED FROM USART.
;ERROR 2: REVR DONE NEVER RECEIVED FROM USART.
;ERROR 3: USART DATA ERROR.

;OTHER:  OTHER DATA PRINTED FOR ERRORS 1 AND 2 INDICATES THE
;        DATA PATTERN THAT WOULD BE WRITTEN TO THE USART IF IT
;        WERE PROPERLY ENABLED.

;DEBUG:  1) CHECK THAT THE MODEM OR TURNAROUND CONNECTOR IS
;           CORRECTLY INSTALLED AND ENABLED.  USE THE ANALOG
;           LOCAL LOOP BACK FEATURE.  THE 'TEST' LAMP SHOULD
;           THEN LIGHT.  USE THE SINGLE LINE EIA TURNAROUND
;           CONNECTOR, H3248, ATTACHED TO THE APT/RD PORT CABLE.
;        2) CHECK THAT THE BAUD RATE OF THE MODEM MATCHES THE
;           BAUD RATE OF THE USART.  THE USART IS EXTERNALLY
;           CLOCKED AT EITHER 300 OR 1200 BAUD.  SWITCH 5 ON
;           WCS E47 SHOULD BE ON FOR 300 BAUD TRANSMISSION, OFF
;           FOR 1200 BAUD TRANSMISSION.  A DF03 MODEM OPERATES
;           AT 300 BAUD IF THE 'HS' BUTTON IS OUT, AND 1200
;           BAUD IF THE 'HS' BUTTON IS IN.  THE 'HS' LAMP SHOULD
;           BE LIT IF THE THE 'HS' BUTTON IS IN, OFF IF OUT.  IF
;           A TURNAROUND CONNECTOR IS BEING USED, THE SWITCH
;           SETTING OF E47 SWITCH 5 DOES NOT MATTER.
;        3) IF THE MACHINE PASSES THE POWER-UP SELFTEST, THEN ERROR
;           3 INDICATES A PROBLEM WITH THE DRIVERS/RECEIVERS OF THE
;           APT/RD USART (E7 OF THE WCS BOARD).  THE INVERTER LOGIC
;           OF PRINTSET WCSN COULD ALSO BE AT FAULT, AS WELL AS THE
;           CABLES AND CONNECTORS TO THE APT/RD PORT.
;
;*****************************************************************************

70CC  465 0000'CD 2F 3E         TEST2F:         HEAD    <^X2F>
          OF 0000'3A
          C9 00D9'D2
          OF 0000'3A
          3D D3 0119'DA
70E2  466
70E2  467 40 DB                                 IN      TRDB            ;CLEAR OUT USART DATA BUFFER
70E4  468
70E4  469 37 3E                                 MVI     A,TRCR_MASK     ;SET UP REMOTE CR REGISTER FOR TEST
70E6  470 43 D3                                 OUT     TRCR
70E8  471
70E8  472 000C'3A                               LDA     ZERO_DAT        ;LOAD THE FIRST DATA PATTERN INTO A
70EB  473
```

```
70EB  474 000D'CD                          CALL    TEST2F_SUB      ;CALL ROUTINE TO WRITE AND READ DATA
70EE  475                                                          ; FROM APT/RD UART
70EE  476
70EE  477 000A'3A                          LDA     ONE_DAT         ;LOAD A DATA PATTERN OF ALL 1'S INTO A
70F1  478
70F1  479 000D'CD                          CALL    TEST2F_SUB      ;CALL ROUTINE TO WRITE/READ DATA
70F4  480
70F4  481
70F4  482
70F4  483 000A'3A                          LDA     ONE_DAT         ;BACKGROUND OF 1'S SET UP
70F7  484 0007'32                          STA     BACKGRND
70FA  485
70FA  486 3C                               INR     A
70FB  487 3C                               INR     A
70FC  488 0008'32                          STA     DATA_PAT        ;FIRST BIT TO TEST
70FF  489
70FF  490 00'07'21                 6$:     LXI     H,BACKGRND
7102  491 AE                               XRA     M               ;GENERATE SHIFTING PATTERNS
7103  492
7103  493 000D'CD                          CALL    TEST2F_SUB      ;CALL WRITE/READ ROUTINE
7106  494
7106  495 0008'3A                          LDA     DATA_PAT        ;GET THE PATTERN USED TO GENERATE DATA
7109  496 07                               RLC                     ;ROTATE LEFT ONE PLACE
710A  497 0008'32                          STA     DATA_PAT        ;MAKE THIS THE NEW PATTERN
710D  498
710D  499 00FF'D2                          JNC     6$              ;KEEP LOOPING FOR ALL 8 BITS
7110  500
7110  501 00'07'21                        LXI     H,BACKGRND
7113  502 34                               INR     M               ;INCREMENT E TO ALL ZERO'S
7114  503
7114  504 00FF'CA                          JZ      6$              ;LOOP FOR 1 FLOATING IN FIELD OF 0'S
7117  505 3C D3                            TAIL
7119  506
7119  507                          ;*****************************************************************************
7119  508                          ;
7119  509                          ;  LAST TEST FOR EVERY CONTOL_SECTION....FORCES END OF SECTION
7119  510                          ;
7119  511                          ;*****************************************************************************
7119  512
7119  513 0000'CD 51 3E            END_TEST:       HEAD    <^X51>
           0F 0000'3A
           C9 0126'D2
           0F 0000'3A
           3D D3 0135'DA
712F  514
712F  515 0F 3E                              MVI     A,15            ;RESET UART
7131  516 43 D3                            OUT     TRCR
7133  517
7133  518 3C D3                            TAIL
7135  519
```

```
$PSW            = 00000006
$SP             = 00000006
A               = 00000007
ALLOWP          = 00000003
APTFLG          = 00000004
AUTO            = 00000005
B               = 00000000
BACKGRND          00000007 R      02
BOOTEN          = 00000007
BOOTF           = 00000001
C               = 00000001
CLRACK          = 000000A9
CLRACL          = 00000033
CLRATN          = 000000AB
CLRBSY          = 0000002E
CLRCLK          = 00000020
CLRCSK          = 00000024
CLRCSR          = 00000038
CLRDCL          = 00000031
CLRHLT          = 000000AF
CLRLIT          = 0000003C
CLRMCI          = 00000029
CLRMCK          = 0000002B
CLRPFI          = 000000AD
CLRSS           = 00000022
CLRTIM          = 000000A5
CLRTMR          = 0000002C
CLRUPC          = 000000A8
CLRUPK          = 00000026
CLRWCK          = 00000036
CNT             = 000007D5
CNTA            = 00000FA0
COMPARE           ********    X    02
CON_ADD_DAT       ********    X    02
CON_BEGIN_TEST    ********    X    02
CON_ERROR         ********    X    02
CON_ERR_NUM       ********    X    02
CON_EXP_DAT       ********    X    02
CON_MOD_DAT       ********    X    02
CON_MSK_DAT       ********    X    02
CON_REC_DAT       ********    X    02
CPATTN          = 00000083
CPUACK          = 00000082
CSR07           = 00000085
CSR15           = 00000086
CSR23           = 00000087
D               = 00000002
DATA_PAT          00000008 R      02
DCLO            = 00000002
DISMEM          = 000000A7
DISPAR          = 000000A1
E               = 00000003
ENBMEM          = 000000A6
ENBPAR          = 000000A0
END_TEST          00000119 R      02
ENTRY             00000000 R      02
EOS_FLG           ********    X    02
ERROR_CON         ********    X    02
EXP_DAT           00000009 R      02
FILE_NAME         00000005 R      02
H               = 00000004
HEAD            = 00000000
HLTFLG          = 00000002
INITH           = 00000035
INITL           = 00000034
ITCSR           = 0000001D
ITDB            = 0000001C
L               = 00000005
LITERAL         = 00000001
LVL             = 00000000
M               = 00000006
MACSTK1         = 000007D5
MISC2           = 00000001
MOVER             ********    X    02
MWCS_A            ********    X    02
NA_EXP_REC        ********    X    02
NA_OTHER          ********    X    02
OKV15           = 00000007
ONE_DAT           0000000A R      02
PARITY_VECTOR   = 00004C22
POWER_VECTOR    = 00004C25
READ            = 00000080
READJ1          = 00000003
READJ2          = 00000004
REC_DAT           0000000B R      02
REMDIS          = 00000006
REPORT_ERR        000000BD R      02
SETACK          = 000000A8
SETACL          = 00000032
SETATN          = 000000AA
SETBSY          = 0000002F
SETCLK          = 00000021
SETCSK          = 00000025
SETCSR          = 00000039
SETDCL          = 00000030
SETHLT          = 000000AE
SETLIT          = 0000003D
SETMCI          = 00000028
SETMCK          = 0000002A
SETPFI          = 000000AC
SETSS           = 00000023
SETTIM          = 000000A4
SETTMR          = 0000002D
SETUPC          = 000000A9
SETUPK          = 00000027
SETWCK          = 00000037
SKIP_TEST         ********    X    02
SYM             = 000007D5
TEST2F            000000CC R      02
TEST2F_ERR1_2     00000083 R      02
TEST2F_ERROR      0000009B R      02
TEST2F_SUB        0000000D R      02
TRCR            = 00000043
TRCR_MASK       = 00000037
TRDB            = 00000040
TRMODE          = 00000042
TRSR            = 00000041
TRSR_MASK       = 00000002
TTCR            = 0000004B
TTDB            = 00000048
TTMODE          = 0000004A
TTSR            = 00000049
TUCR            = 00000047
TUDB            = 00000044
TUMODE          = 00000046
TUSR            = 00000045
UPC14           = 00000084
UPC14A          = 00000000
VERSION           00000003 R      02
VNORML          = 000000A3
VSHIFT          = 000000A2
WCSB0           = 00000008
WCSB1           = 00000009
WCSB2           = 0000000A
WRITE           = 000000CC
Y               = 00000042
YBUSRD          = 000000EC
ZERO_DAT          0000000C R      02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
CPU6	00000135 (309.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	26	00:00:00.12	00:00:02.45
Command processing	37	00:00:00.28	00:00:07.95
Pass 1	385	00:00:06.17	00:01:37.99
Symbol table sort	0	00:00:00.15	00:00:01.09
Pass 2	151	00:00:03.06	00:00:34.49
Symbol table output	14	00:00:00.10	00:00:00.99
Psect synopsis output	2	00:00:00.02	00:00:00.18
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	618	00:00:09.92	00:02:25.15

The working set limit was 715 pages.
39917 bytes (78 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 138 non-local and 11 local symbols.
2474 source lines were read in Pass 1, producing 14 object records in Pass 2.
148 pages of virtual memory were used to define 147 macros.

Macro library statistics

Macro library name	Macros defined
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

/SHOW=(BIN)/LIST=ENKBG.LIS/OBJECT=ENKBG.OBJ 8085MAC.MAR+MACDEF.MAC+[]ENKBG.MAC